



# SOFTWARE ENGINEERING

C03001

## CHAPTER 9 – SOFTWARE EVOLUTION

# TOPICS COVERED

- ✓ Evolution processes
- ✓ Legacy systems
- ✓ Software maintenance



# SOFTWARE CHANGE

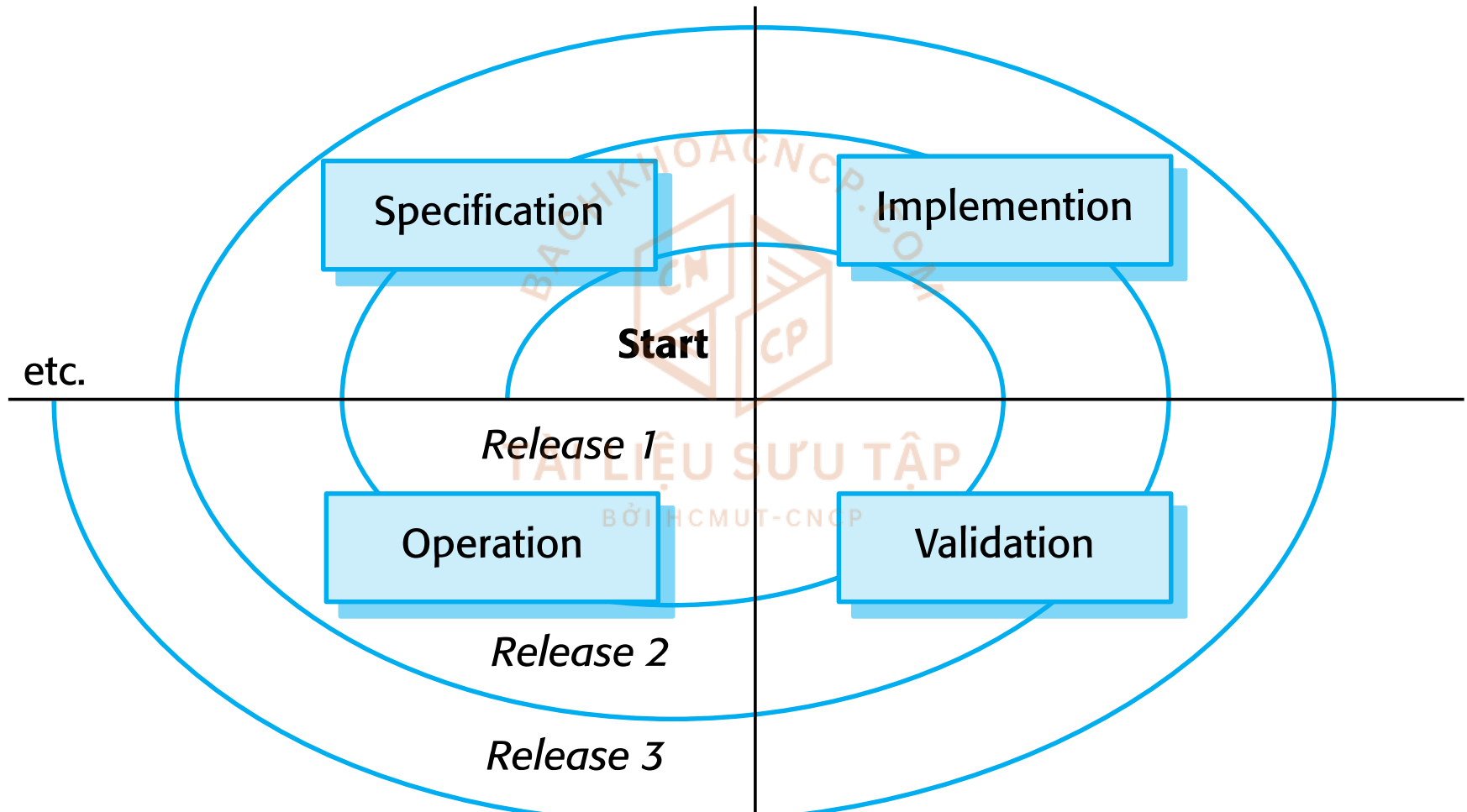
- ✓ Software change is inevitable
  - New requirements emerge when the software is used;
  - The business environment changes;
  - Errors must be repaired;
  - New computers and equipment is added to the system;
  - The performance or reliability of the system may have to be improved.
- ✓ A key problem
  - implementing and managing change to their existing software systems.

# IMPORTANCE OF EVOLUTION

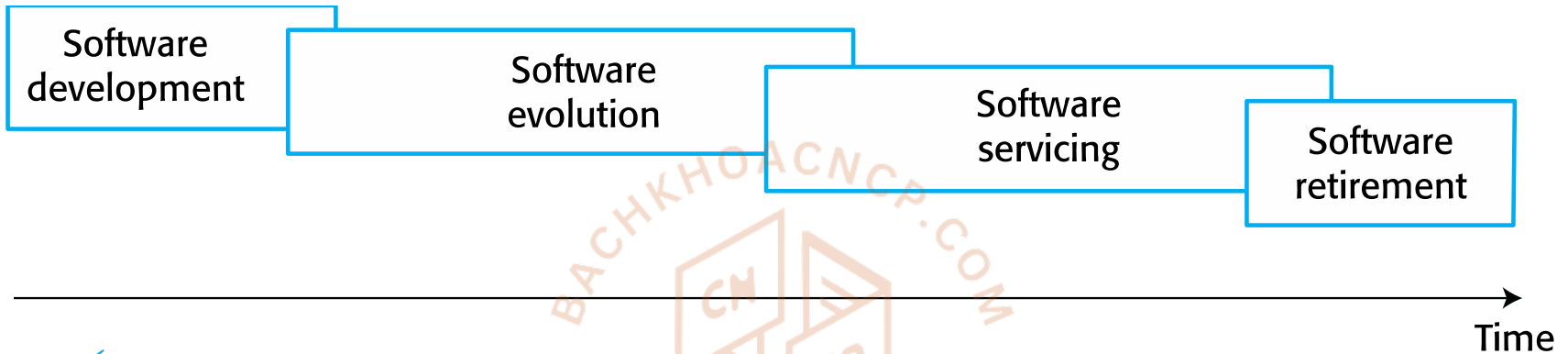
- ✓ Organisations have huge investments in their software systems - they are critical business assets.
- ✓ To maintain the value of these assets to the business, they must be changed and updated.

*The majority of the software budget in large companies is devoted to changing and evolving existing software rather than developing new software.*

# A SPIRAL MODEL OF DEVELOPMENT AND EVOLUTION



# EVOLUTION AND SERVICING



## ✓ Evolution

- in operational use and is evolving as new requirements are proposed and implemented in the system.

## ✓ Servicing

- the software remains useful but the only changes made are those required to keep it operational i.e. bug fixes and changes to reflect changes in the software's environment.

## ✓ Phase-out

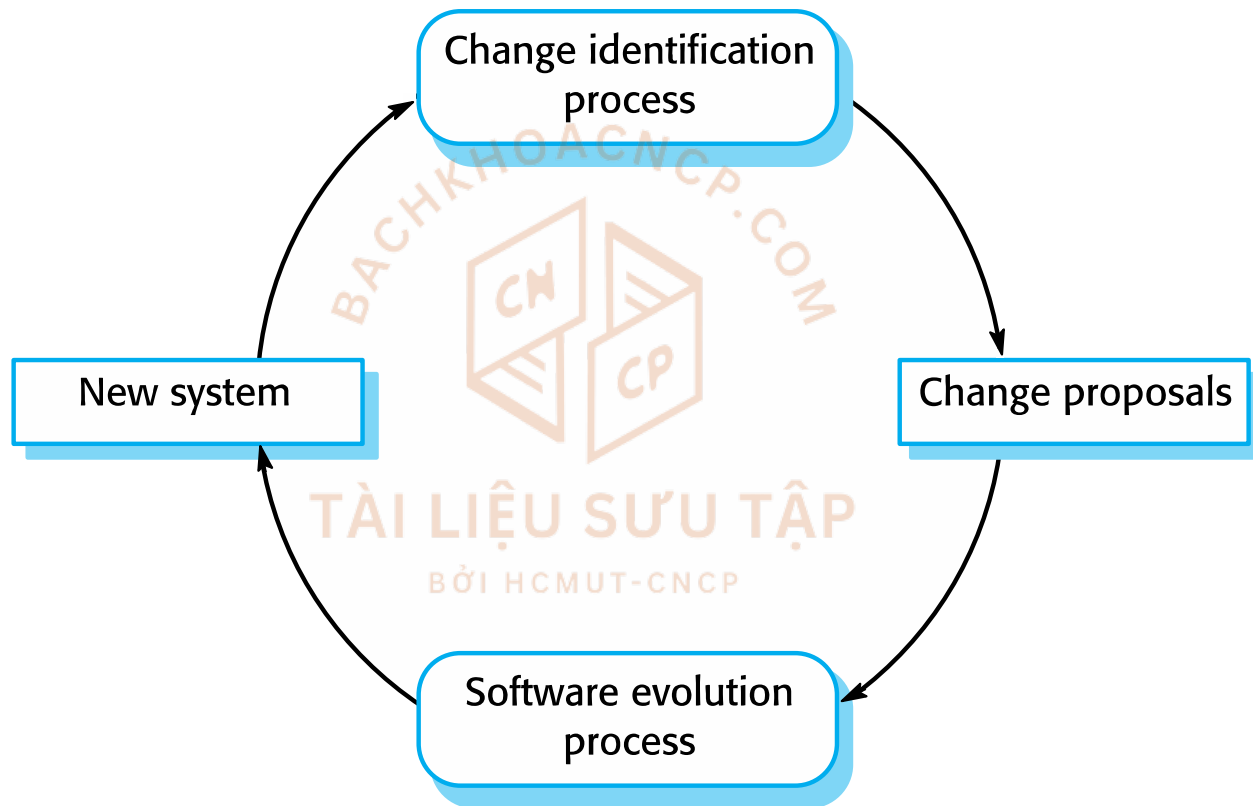
- The software may still be used but no further changes are made



TÀI LIỆU SƯU TẬP

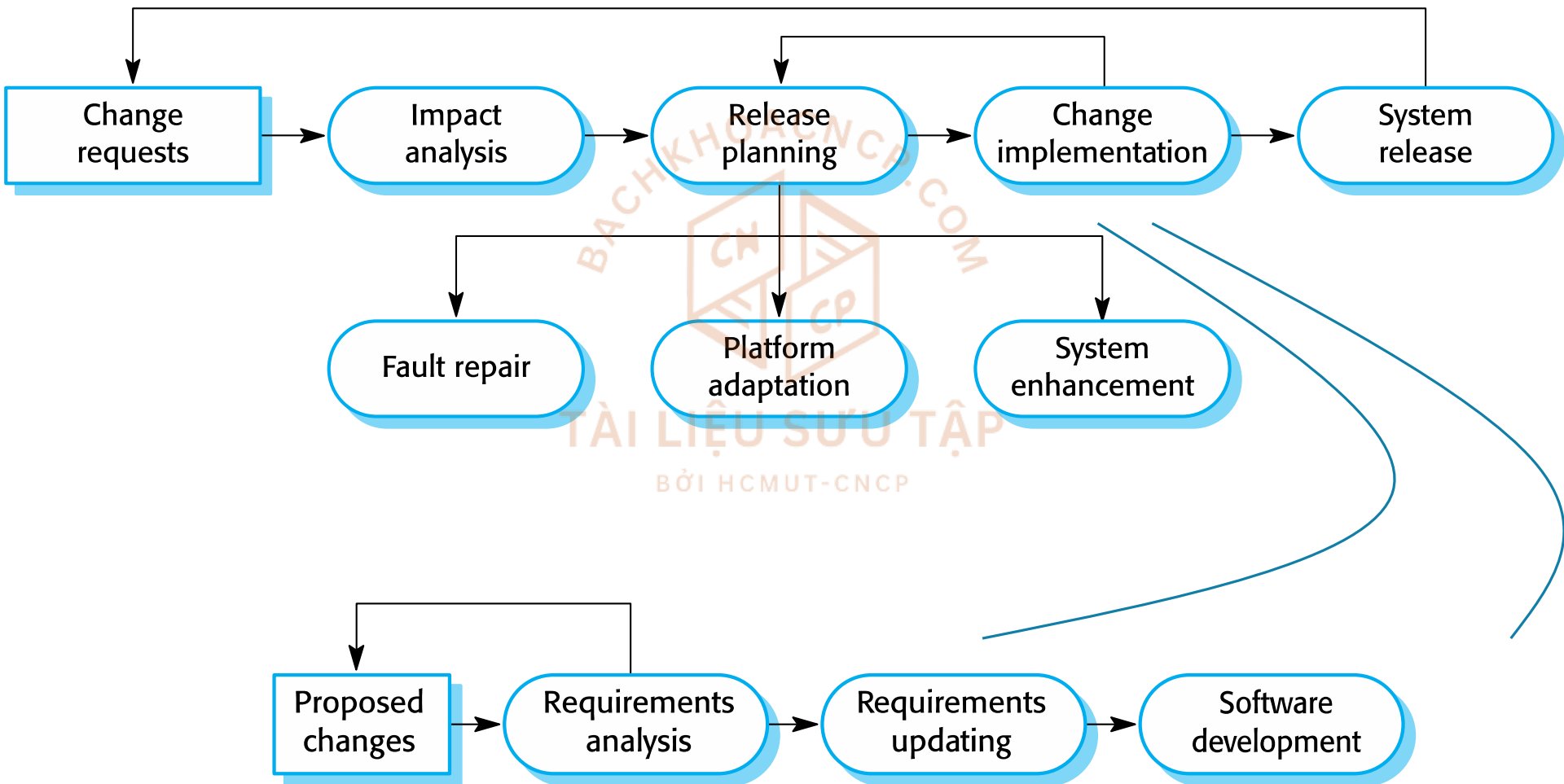
# EVOLUTION PROCESSES

# CHANGE IDENTIFICATION AND EVOLUTION PROCESSES

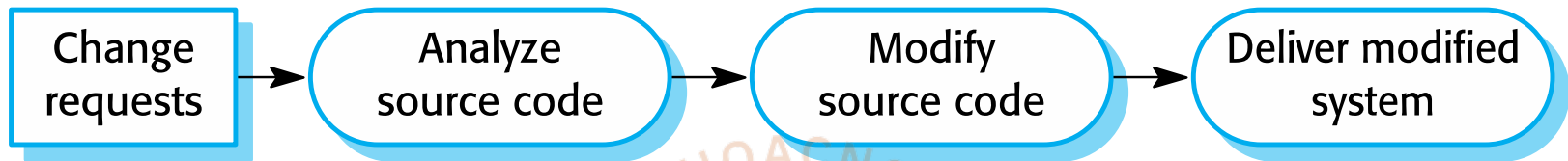




# THE SOFTWARE EVOLUTION PROCESS




# URGENT CHANGE REQUESTS



- ✓ Urgent changes may have to be implemented without going through all stages of the software engineering process
  - If a serious system fault has to be repaired to allow normal operation to continue;
  - If changes to the system's environment (e.g. an OS upgrade) have unexpected effects;
  - If there are business changes that require a very rapid response (e.g. the release of a competing product).



BACHKHOACNCP.COM



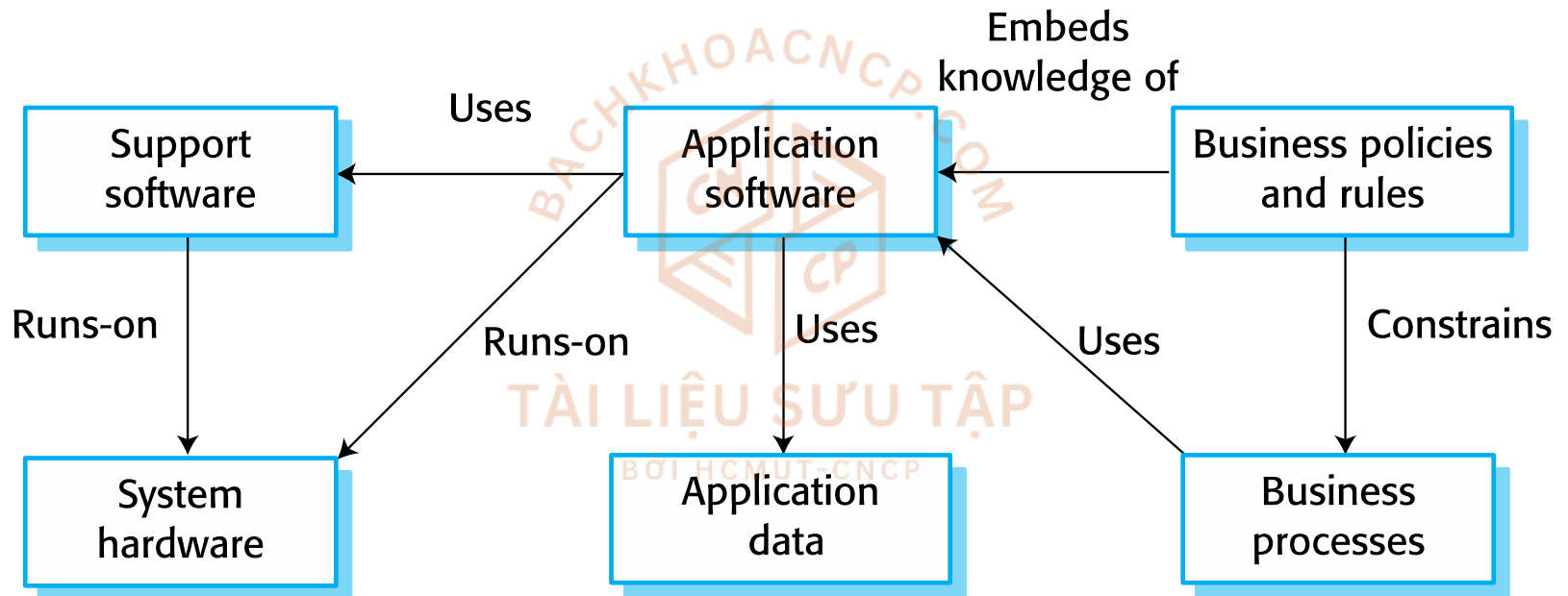
TÀI LIỆU SƯU TẬP  
BỘ MÔN CÔNG NGHỆ THÔNG TIN

# LEGACY SYSTEMS

# LEGACY SYSTEMS

- ✓ Legacy systems are older systems that rely on languages and technology that are no longer used for new systems development.
- ✓ Legacy software may be dependent on older hardware, such as mainframe computers and may have associated legacy processes and procedures.
- ✓ Legacy systems are not just software systems but are broader socio-technical systems that include hardware, software, libraries and other supporting software and business processes.

# THE ELEMENTS OF A LEGACY SYSTEM



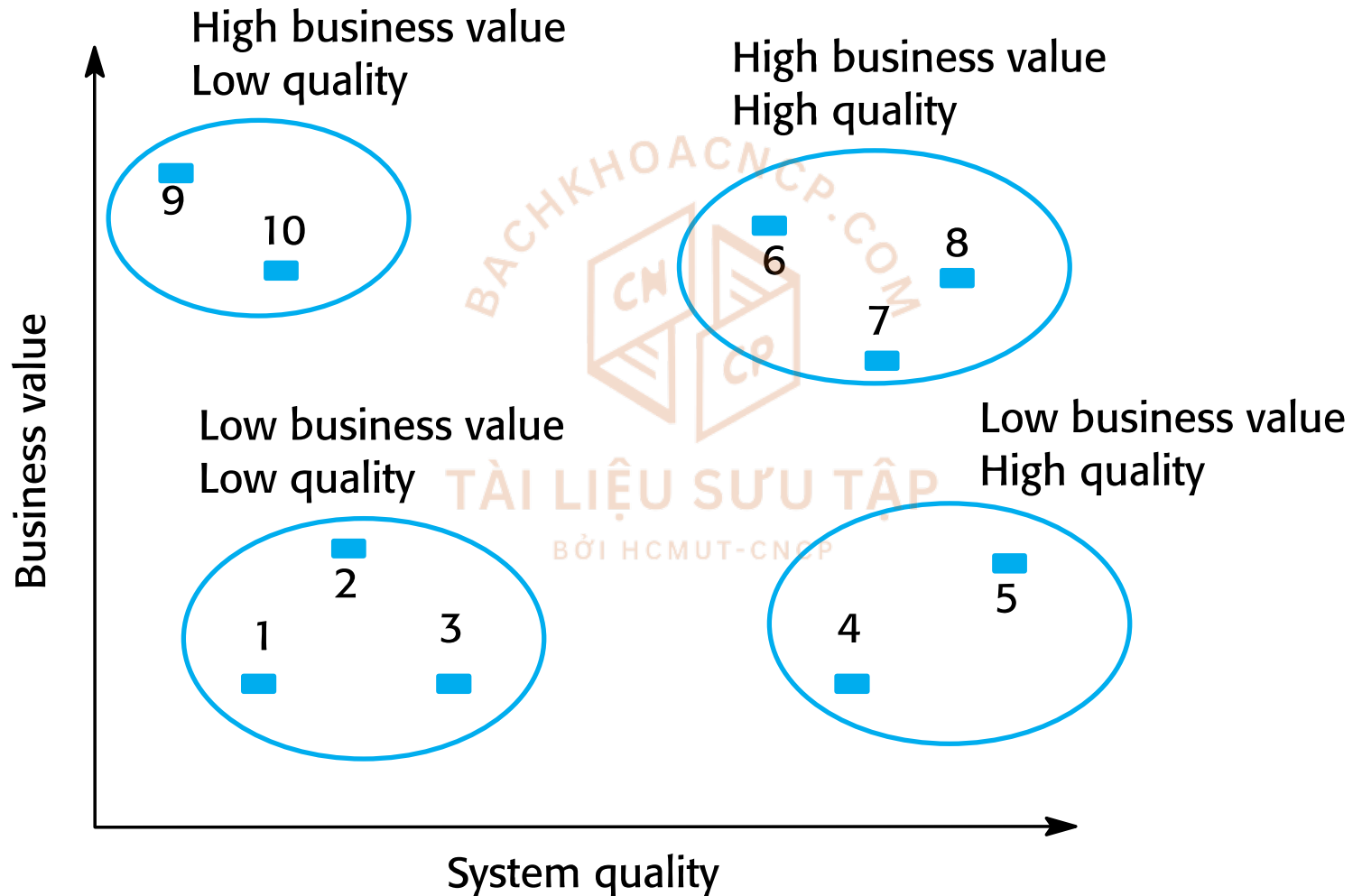
# LEGACY SYSTEM REPLACEMENT, CHANGE, MANAGEMENT

- ✓ Legacy system replacement is risky and expensive
  - Lack of complete system specification
  - Tight integration of system and business processes
  - Undocumented business rules embedded in the legacy system
  - New software development may be late and/or over budget
- ✓ Legacy systems are expensive to change
  - No consistent programming style
  - Use of obsolete programming languages with few people available with these language skills
  - Inadequate system documentation
  - System structure degradation
  - Program optimizations may make them hard to understand
  - Data errors, duplication and inconsistency

# LEGACY SYSTEM MANAGEMENT

- ✓ Organisations that rely on legacy systems must choose a strategy for evolving these systems
  - Scrap the system completely and modify business processes so that it is no longer required;
  - Continue maintaining the system;
  - Transform the system by re-engineering to improve its maintainability;
  - Replace the system with a new system.
- ✓ The strategy chosen should depend on the system quality and its business value.

# FIGURE 9.13 AN EXAMPLE OF A LEGACY SYSTEM ASSESSMENT





# LEGACY SYSTEM CATEGORIES

- ✓ Low quality, low business value
  - These systems should be scrapped.
- ✓ Low-quality, high-business value
  - These make an important business contribution but are expensive to maintain. Should be re-engineered or replaced if a suitable system is available.
- ✓ High-quality, low-business value
  - Replace with COTS, scrap completely or maintain.
- ✓ High-quality, high business value
  - Continue in operation using normal system maintenance.



TÀI LIỆU SƯU TẬP

# SOFTWARE MAINTENANCE

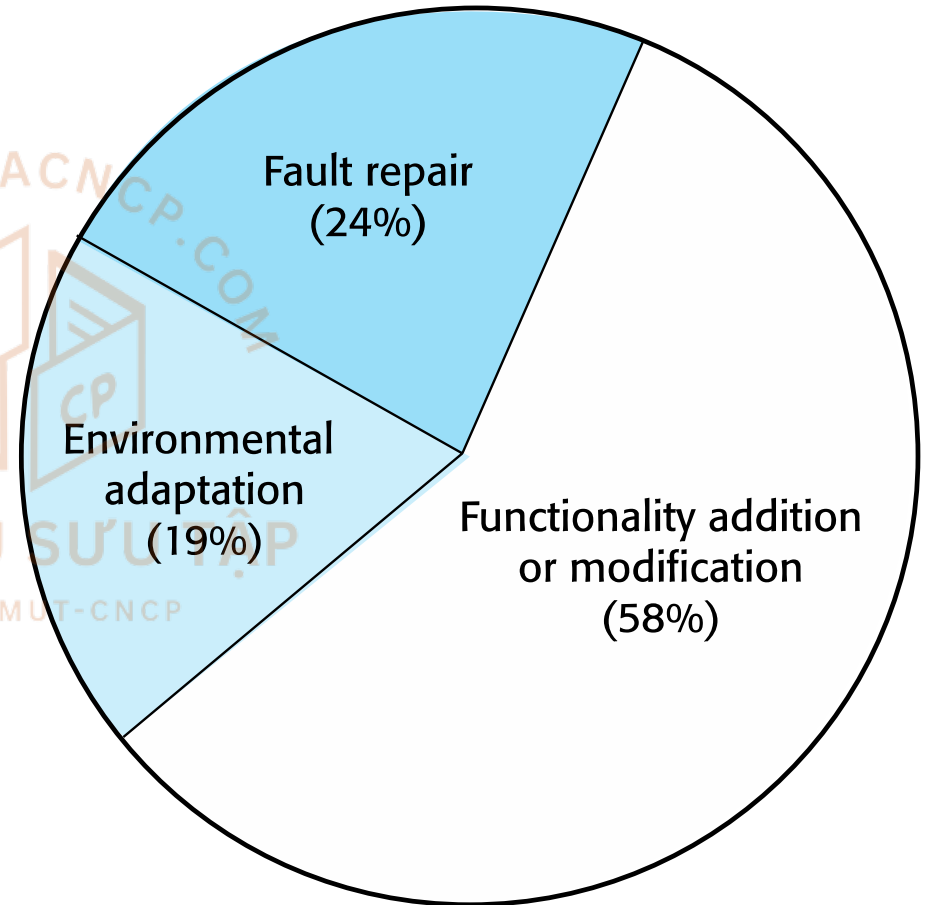


# SOFTWARE MAINTENANCE

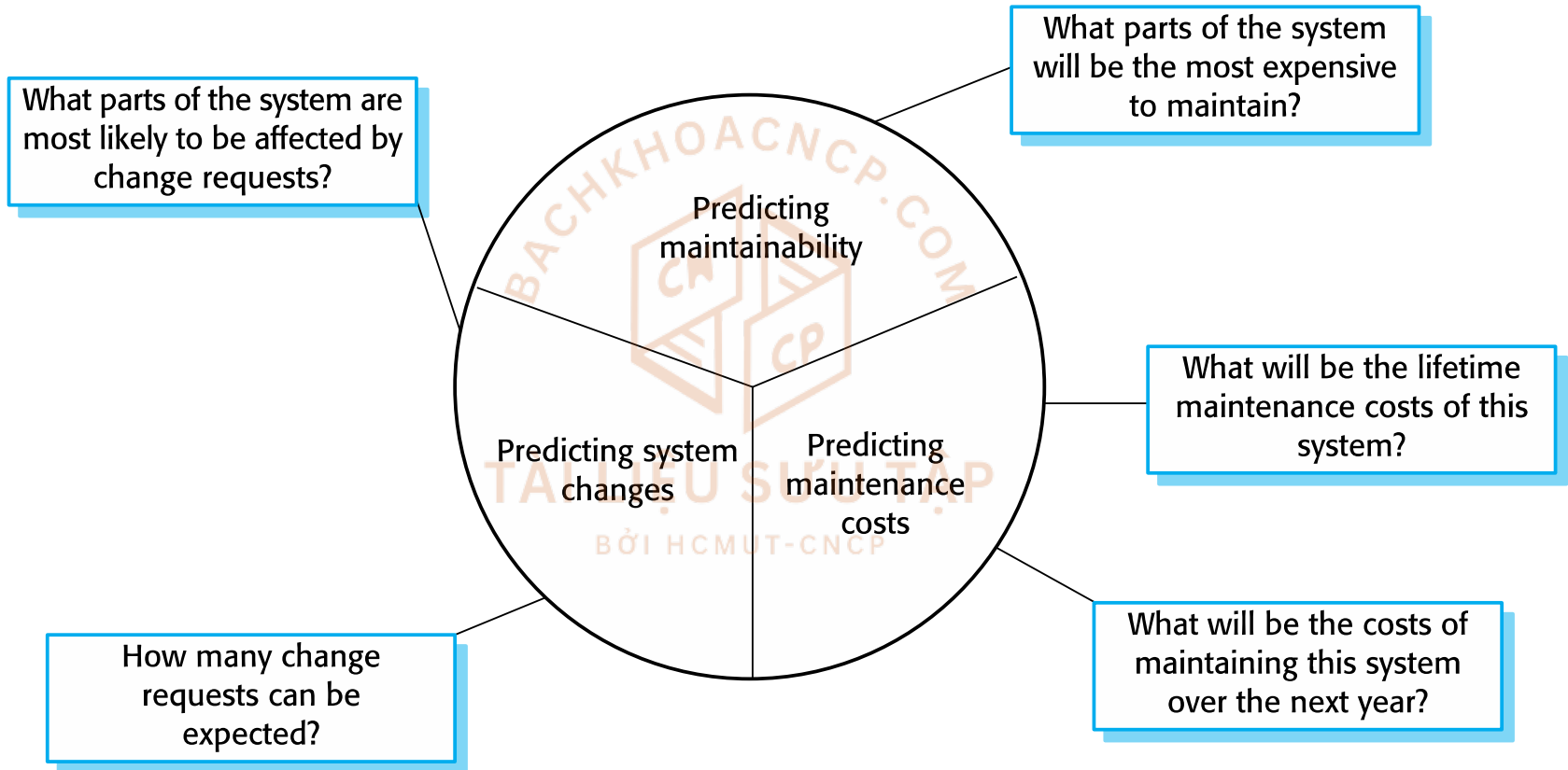
- ✓ Modifying a program after it has been put into use.
- ✓ The term is mostly used for changing custom software. Generic software products are said to evolve to create new versions.
- ✓ Maintenance does not normally involve major changes to the system's architecture.
- ✓ Changes are implemented by modifying existing components and adding new components to the system.

# TYPES OF MAINTENANCE

- ✓ Maintenance to repair software faults
- ✓ Maintenance to adapt software to a different operating environment
- ✓ Maintenance to add to or modify the system's functionality



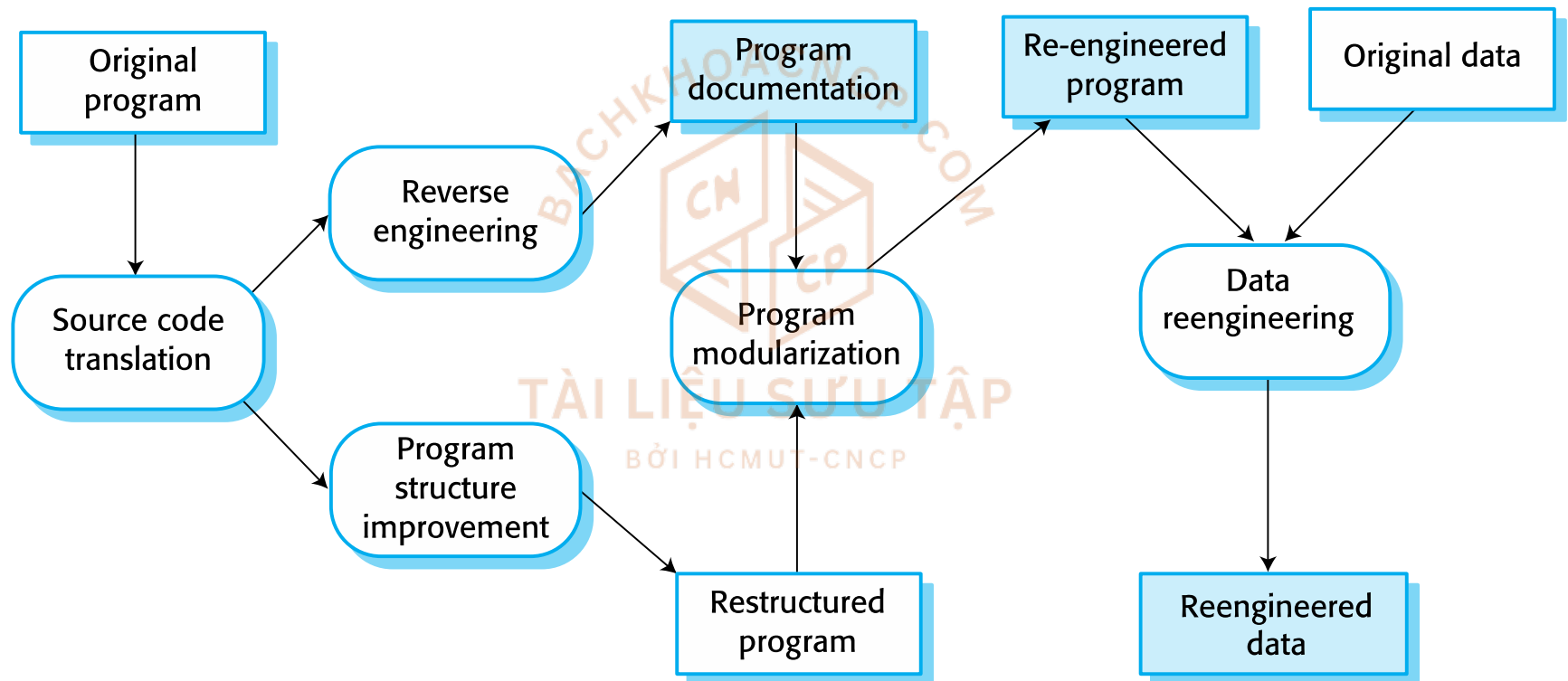
# MAINTENANCE PREDICTION



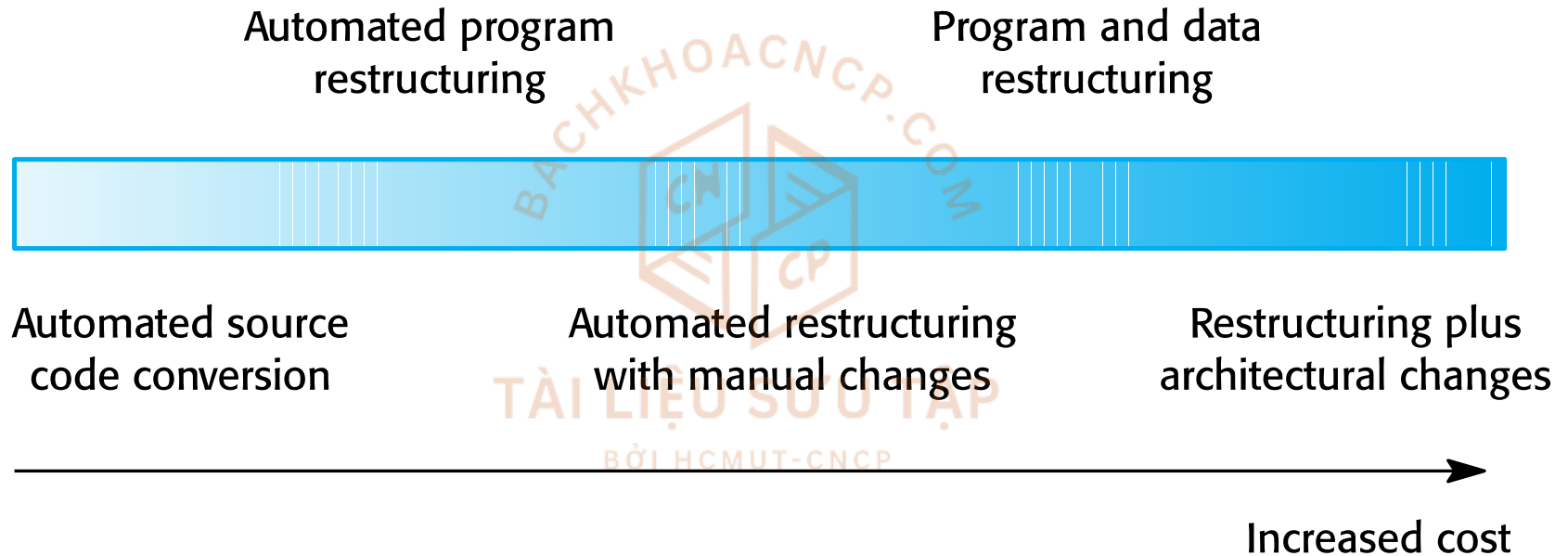
# SYSTEM RE-ENGINEERING

- ✓ Re-structuring or re-writing part or all of a legacy system without changing its functionality.
- ✓ Applicable where some but not all sub-systems of a larger system require frequent maintenance.
- ✓ Re-engineering involves adding effort to make them easier to maintain. The system may be re-structured and re-documented.

# THE REENGINEERING PROCESS



# REENGINEERING APPROACHES





# REFACTORING

- ✓ Refactoring => making improvements
  - to slow down degradation through change.
  - 'preventative maintenance' => reduces the problems of future change.
- ✓ Refactoring => modifying programs
  - to improve structure, reduce complexity, for easier to understand.
- ✓ Concentrate on program improvement.

# REFACTORING AND REENGINEERING

- ✓ Re-engineering takes place after a system has been maintained for some time and maintenance costs are increasing. You use automated tools to process and re-engineer a legacy system to create a new system that is more maintainable.
- ✓ Refactoring is a continuous process of improvement throughout the development and evolution process. It is intended to avoid the structure and code degradation that increases the costs and difficulties of maintaining a system.

# 'BAD SMELLS' IN PROGRAM CODE

- ✓ Duplicate code
- ✓ Long methods
- ✓ Switch (case) statements
- ✓ Data clumping
  - Data clumps occur when the same group of data items (fields in classes, parameters in methods) re-occur in several places in a program. These can often be replaced with an object that encapsulates all of the data.
- ✓ Speculative generality
  - This occurs when developers include generality in a program in case it is required in the future. This can often simply be removed.

# SUMMARY

- ✓ Development and evolution can be integrated, iterative
- ✓ The costs of maintenance usually exceed the development costs
- ✓ Software evolution is driven by
  - requests for changes; process: change impact analysis, release planning and change implementation.
- ✓ 3 types of software maintenance
  - bug fixing, modifying to new environment, new or changed requirements.
- ✓ Software re-engineering
  - re-structuring and re-documenting software: easier to understand and change.
- ✓ Refactoring
  - changes: preserve functionality; a preventative maintenance.
- ✓ Legacy system
  - Business value vs. quality